Master Method

Proof (Part I)

Design and Analysis
of Algorithms I

The Master Method

If $T(n) \leq aT\left(\frac{n}{b}\right) + O(n^d)$

then

$$T(n) = \begin{cases} O(n^d \log n) & \text{if } a = b^d \quad \text{(Case 1)} \\ O(n^d) & \text{if } a < b^d \quad \text{(Case 2)} \\ O(n^{\log_b a}) & \text{if } a > b^d \quad \text{(Case 3)} \end{cases}$$

Preamble

Assume: recurrence is

(i) $T(1) \leq c$

(ii) $T(n) \leq aT(\frac{n}{b}) + cn^d$ (for some constant c)

And n is a power of b.

(general case is similar, but more tedious)

Idea: generalize MergeSort analysis.

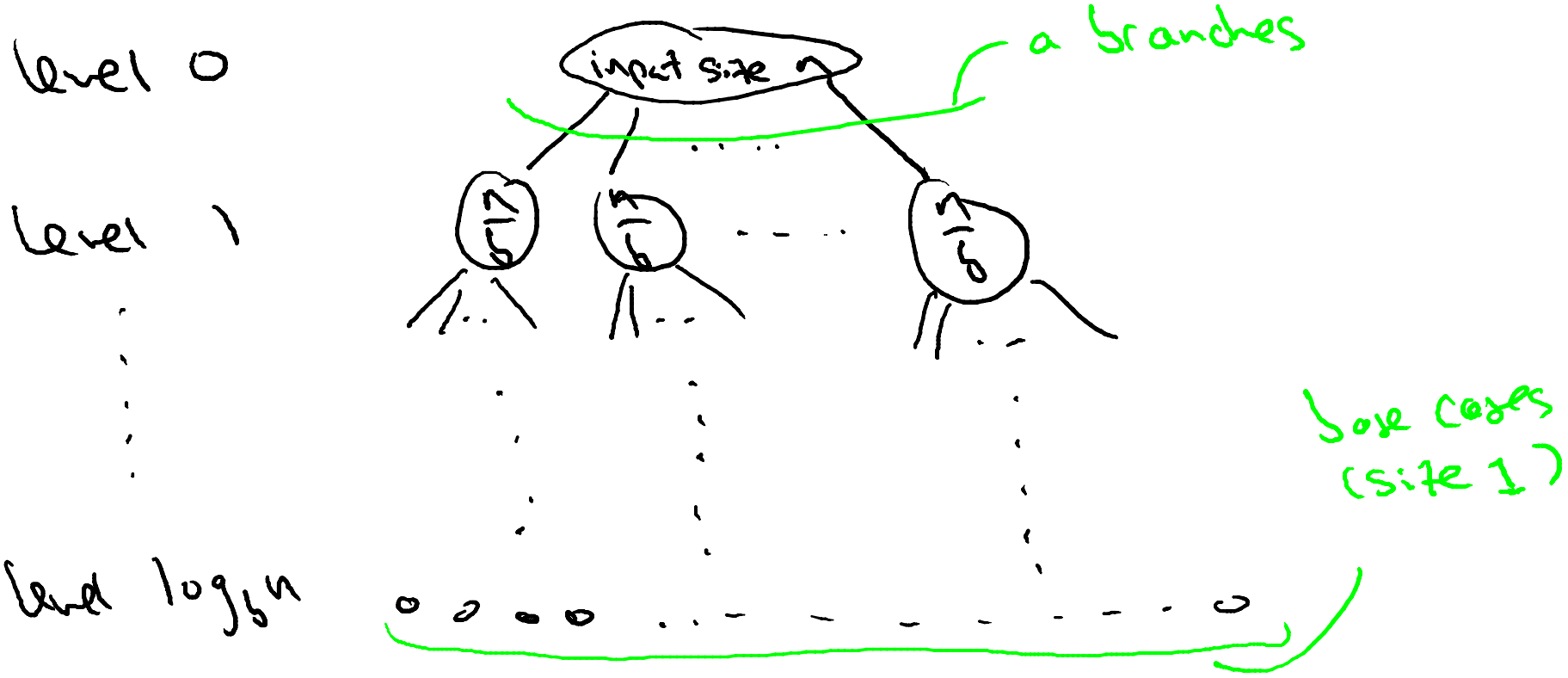
(i.e., use a recursion tree)

What is the pattern? Fill in the blanks in the following statement: at each level j=0,1,2,…, $\log_b n$, there are <*blank*> subproblems, each of size <*blank*>.

→ # of times you can divide n by b before reaching 1

- ○ a^j and n/a^j, respectively.
- ○ a^j and n/b^j, respectively.
- ○ b^j and n/a^j, respectively.
- ○ b^j and n/b^j, respectively.

The Recursion Tree

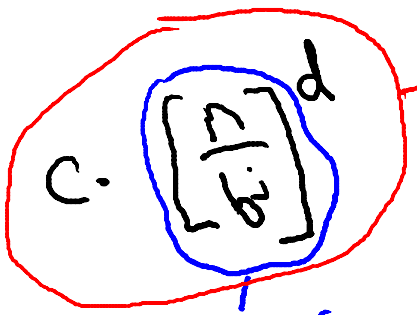


Tim Roughgarden

Work at a Single Level

Total work at level j [ignoring work in recursive calls]

$$\leq a^j \cdot c \cdot \left[\frac{n}{b^j}\right]^d = cn^d \cdot \left[\frac{a}{b^d}\right]^j$$



# of level-j subproblems

size of each level-j subproblem

Total Work

Summing over all levels $j = 0, 1, 2, \ldots, \log_b n$:

$$\text{total work} \leq \boxed{cn^d \cdot \sum_{j=0}^{\log_b n} \left[\frac{a}{b^d}\right]^j} \quad (*)$$